# 准备

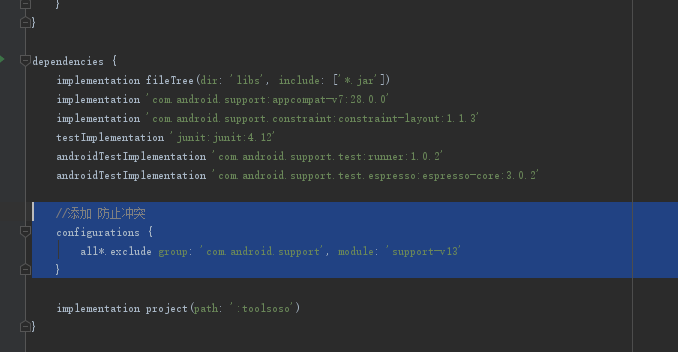
工程的build文件

allprojects添加maven { url "https://jitpack.io" }



app 的build文件 dependencies标签下添加

//添加 防止冲突  
configurations {  
 all\*.exclude group: 'com.android.support', module: 'support-v13'  
}

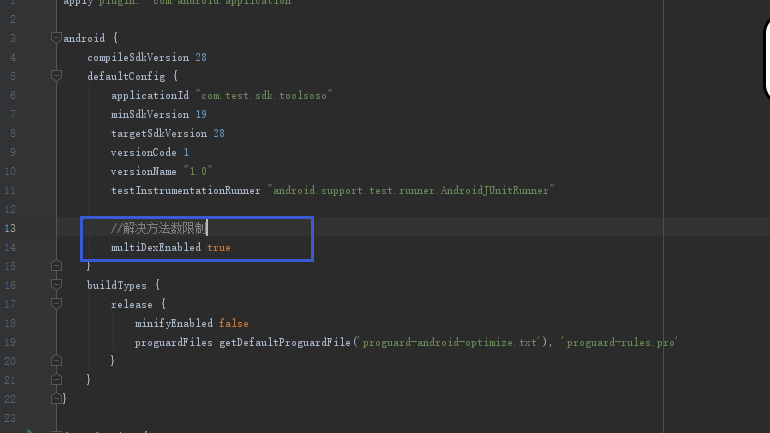


# CheckUtil 检测app是否为debug版本

*app* = this;  
boolean isdebug = CheckUtil.*isApkInDebug*(this);

# multidex解决方法数限制性

app的build 文件下 defaultConfig标签下添加 multiDexEnabled true



# 腾讯x5

清单文件添加：

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

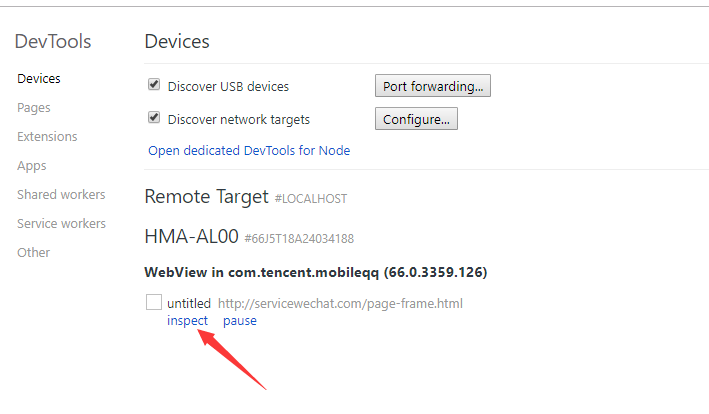
# Stetho 基于谷歌浏览器的快捷调试

初始化

Stetho.*initializeWithDefaults*(this);

应用跑起来后 谷歌浏览器打开<chrome://inspect/#devices>

点击inspect进入即可



# Bugly 异常上报到平台

初始化 AppConfig.*BuglyAppId*;为您在bugly平台注册的应用后分配的appid

String buglyId = AppConfig.*BuglyAppId*;  
String CrashChannel = BuglyCrash.*DebugChannel*;  
BuglyCrash.*InitCrash*(this, buglyId, CrashChannel, true);

上报异常

BuglyCrash.*PostCatchedException*(e);

# Log 日志操作工具

初始化

FormatStrategy formatStrategy = PrettyFormatStrategy.*newBuilder*()  
 .showThreadInfo(false) //是否选择显示线程信息，默认为true  
 .methodCount(0) //方法数显示多少行，默认2行  
 .methodOffset(7) //隐藏方法内部调用到偏移量，默认5  
 .tag(AppConfig.*TAG*) //自定义TAG全部标签，默认PRETTY\_LOGGER  
 .build();  
LogUtil.*Init*(formatStrategy, isDebug);

# Router 路由操作

app的build文件 defaultConfig标签下添加

//arouter 路由  
javaCompileOptions {  
 annotationProcessorOptions {  
 arguments = [moduleName: project.getName()]  
 }  
}

初始化

if (b) {  
 ARouter.*openLog*(); // 打印日志  
 ARouter.*openDebug*(); // 开启调试模式(如果在InstantRun模式下运行，必须开启调试模式！线上版本需要关闭,否则有安全风险)  
}  
ARouter.*init*(this); // 尽可能早，推荐在Application中初始化

# Xutils 简单工具类

初始化

x.Ext.*init*(this);  
x.Ext.*setDebug*(b); // 是否输出debug日志, 开启debug会影响性能.

# Blankj 常用工具

初始化

Utils.*init*(this);

# Leakcanary 内存泄漏检测

初始化

private RefWatcher refWatcher;  
public static RefWatcher getRefWatcher(Context context) {  
 LandscapeApp application = (LandscapeApp) context.getApplicationContext();  
 return application.refWatcher;  
}  
private void InitLeakCanary() {  
 refWatcher = LeakManager.*Init*(this);  
}